



## IMPROVED ARCHITECTURE WITH SHARED MEMORY

### Field of the Invention

The present invention relates generally to integrated circuits (ICs). More particularly, the invention relates to an improved architecture with shared memory.

### Background of the Invention

Fig. 1 shows a block diagram of a portion of a conventional System-on-Chip (SOC) 100, such as a digital signal processor (DSP). As shown, the SOC includes a processor 110 coupled to a memory module 160 via a bus 180. The memory module stores a computer program comprising a sequence of instructions. During operation of the SOC, the processor retrieves and executes the computer instructions from memory to perform the desired function.

An SOC may be provided with multiple processors that execute, for example, the same program. Depending on the application, the processors can execute different programs or share the same program. Generally, each processor is associated with its own memory module to improve performance because a memory module can only be accessed by one processor during each clock cycle. Thus, with its own memory, a processor need not wait for memory to be free since it is the only processor that will be accessing its

associated memory module. However, the improved performance is achieved at the sacrifice of chip size since duplicate memory modules are required for each processor.

As evidenced from the above discussion, it is  
5 desirable to provide systems in which the processors can share a memory module to reduce chip size without incurring the performance penalty of conventional designs.

Summary of the Invention

10 The invention relates, in one embodiment, to a method of sharing a memory module between a plurality of processors. The memory module is divided into n banks, where n = at least 2. Each bank can be accessed by one or more processors at any one time. The memory module is  
15 mapped to allocate sequential addresses to alternate banks of the memory, where sequential data are stored in alternate banks due to the mapping of the memory. In one embodiment, the memory banks are divided into x blocks, where x = at least 1, wherein each block can be accessed by  
20 one of the plurality of processors at any one time. In another embodiment, the method further includes synchronizing the processors to access different blocks at any one time. In another embodiment, first and second signal paths are provided between the memory module and a  
25 processor. The first signal path couples a cache to a

processor and memory module for enabling the processor to fetch a plurality of data words from different banks simultaneously. This reduces memory latency caused by memory contention. The second signal path couples the 5 memory module directly to the processor.

**Brief Description of the Drawings**

Fig. 1 shows a block diagram of conventional SOC;

Fig. 2 shows a system in accordance with one 10 embodiment of the invention;

Figs. 3-5 show a flow of FCU in accordance with different embodiments of the invention;

Fig. 6 shows a system in accordance with another embodiment of the invention;

15 Figs. 7-8 show flow diagrams of an arbitration unit in accordance with various embodiments of the invention; and

Figs. 9-10 show memory modules in accordance with various embodiments of the invention.

20 **Preferred Embodiments of the Invention**

Fig. 2 shows a block diagram of a portion of a system 200 in accordance with one embodiment of the invention.

The system comprises, for example, multiple digital signal processors (DSPs) for multi-port digital subscriber line 25 (DSL) applications on a single chip. The system comprises

m processors 210, where m is a whole number equal to or greater than 2. Illustratively, the system comprises first and second processors 210a-b ( $m = 2$ ). Providing more than two processors in the system is also useful.

5       The processors are coupled to a memory module 260 via respective memory buses 218a and 218b. The memory bus, for example, is 16 bits wide. Other size buses can also be used, depending on the width of each data byte. Data bytes accessed by the processors are stored in the memory module.

10      In one embodiment, the data bytes comprise program instructions, whereby the processors fetch instructions from the memory module for execution.

      In accordance with one embodiment of the invention, the memory module is shared between the processors without 15 noticeable performance degradation, eliminating the need to provide duplicate memory modules for each processor.

      Noticeable performance degradation is avoided by separating the memory module into n number of independently operable banks 265, where n is an integer greater than or equal to 20 2. Preferably,  $n =$  the number of processors in the system (i.e.  $n = m$ ). Since the memory banks operate independently, processors can simultaneously access different banks of the memory module during the same clock cycle.

In another embodiment, a memory bank is subdivided into x number of independently accessible blocks 275a-p, where x is an integer greater than or equal to 1. In one embodiment, each bank is subdivided into 8 independently accessible blocks. Generally, the greater the number of blocks, the lower the probability of contention. The number of blocks, in one embodiment, is selected to optimize performance and reduce contention.

In one embodiment, each processor (210a or 210b) has a bus (218a or 218b) coupled to each bank. The blocks of the memory array, each have, for example control circuitry 278 to appropriately place data on the bus to the processors. The control circuitry comprises, for example, multiplexing circuitry or tri-state buffers to direct the data to the right processor. Each bank, for example, is subdivided into 8 blocks. By providing independent blocks within a bank, processors can advantageously access different blocks, irrespective of whether they are from the same bank or not. This further increases system performance by reducing potential conflicts between processors.

Furthermore, the memory is mapped so that contiguous memory addresses are rotated between the different memory banks. For example, in a two-bank memory module (e.g., bank 0 and bank 1), one bank (bank 0) would be assigned the even addresses while odd addresses are assigned to the

other bank (bank 1). This would result in data bytes in sequential addresses being stored in alternate memory banks, such as data byte 1 in bank 0, data byte 2 in bank 1, data byte 3 in bank 0 and so forth. The data bytes, in one embodiment, comprise instructions in a program. Since program instructions are executed in sequence with the exception of jumps (e.g., branch and loop instructions), a processor would generally access different banks of the memory module after each cycle during program execution.

- 10 By synchronizing or staggering the processors to execute the program so that the processors access different memory banks in the same cycle, multiple processors can execute the same program stored in memory module 260 simultaneously.

- 15 A flow control unit (FCU) 245 synchronizes the processors to access different memory blocks to prevent memory conflicts or contentions. In the event of a memory conflict (e.g. two processors accessing the same block simultaneously), the FCU locks one of the processors (e.g. inserts a wait state or cycle) while allowing the other processor to access the memory. This should synchronize the processors to access different memory banks in the next clock cycle. Once synchronized, both processors can access the memory module during the same clock cycle until a 20 memory conflict caused by, for example, a jump instruction,

occurs. If both processors (210a and 210b) tries to access block 275a in the same cycle, a wait state is inserted in, for example, processor 210b for one cycle, such that processor 210a first accesses block 275a. In the next 5 clock cycle, processor 210a accesses block 275b and processor 210b accesses block 275a. The processors 210a and 210b are hence synchronized to access different memory banks in the subsequent clock cycles.

Optionally, the processors can be provided with 10 respective critical memory modules 215. The critical memory module, for example, is smaller than the main memory module 260 and is used for storing programs or subroutines which are accessed frequently by the processors (e.g., MIPS critical). The use of critical memory modules enhances 15 system performance by reducing memory conflicts without going to the extent of significantly increasing chip size. A control circuit 214 is provided. The control circuit is coupled to bus 217 and 218 to appropriately multiplex data from memory module 260 or critical memory module 215. In 20 one embodiment, the control circuit comprises tri-state buffers to decouple and couple the appropriate bus to the processor.

In one embodiment, the FCU is implemented as a state machine. Fig. 3 shows a general process flow of a FCU 25 state machine in accordance with one embodiment of the

invention. As shown, the FCU controls accesses by the processors (e.g., A or B). At step 310, the FCU is initialized. During operation, the processors issue respective memory addresses ( $A_{Add}$  or  $B_{Add}$ ) corresponding to 5 the memory access in the next clock cycle. The FCU compares  $A_{Add}$  and  $B_{Add}$  at step 320 to determine whether there is a memory conflict or not (e.g., whether the processors are accessing the same or different memory blocks). In one embodiment, the FCU checks the addresses to determine if 10 any critical memory modules are accessed (not shown). If either processor A or processor B is accessing its respective local critical memory, no conflict occurs.

If no conflict exists, the processors access the memory module at step 340 in the same cycle. If a conflict 15 exists, the FCU determines the priority of access by the processors at step 350. If processor A has a higher priority, the FCU allows processor A to access the memory while processor B executes a wait state at step 360. If processor B has a higher priority, processor B accesses the 20 memory while processor A executes a wait state at step 370. After step 340, 360, or 370, the FCU returns to step 320 to compare the addresses for the next memory access by the processors. For example, if a conflict exists, such as at step 360, a wait state is inserted for processor B while 25 processor A accesses the memory at address  $A_{Add}$ . Hence,

both processors are synchronized to access different memory blocks in subsequent cycles.

Fig. 4 shows a process flow 401 of an FCU in accordance with another embodiment of the invention. In 5 the case of a conflict, the FCU assigns access priority at step 460 by examining processor A to determine whether it has executed a jump or not. In one embodiment, if processor B has executed a jump, then processor B is locked (e.g. a wait state is executed) while processor A is 10 granted access priority. Otherwise, processor A is locked and processor B is granted access priority.

In one embodiment, the FCU compares the addresses of processor A and processor B in step 440 to determine if the processors are accessing the same memory block. In the 15 event that the processors are accessing different memory blocks (i.e., no conflict), the FCU allows both processors to access the memory simultaneously at step 430. If a conflict exists, the FCU compares, for example, the least significant bits of the current and previous addresses of 20 processor A to determine access priority in step 460. If the least significant bits are not equal (i.e. the current and previous addresses are consecutive), processor B may have caused the conflict by executing a jump. As such, the FCU proceeds to step 470, locking processor B while 25 allowing processor A to access the memory. If the least

significant bits are equal, processor A is locked and processor B accesses the memory at step 480.

Fig. 5 shows an FCU 501 in accordance to an alternative embodiment of the invention. Prior to 5 operation, the FCU is initialized at step 510. At step 520, the FCU compares the addresses of processors to determine if they access different memory blocks. If the processors are accessing different memory blocks, both processors are allowed access at step 530. However, if the 10 processors are accessing the same memory block, a conflict exists. During a conflict, the FCU determines which of the processors caused the conflict, e.g., performed a jump. In one embodiment, at steps 550 and 555, the least significant bits of the current and previous addresses of the 15 processors are compared. If processor A caused the jump (e.g., least significant bits of previous and current address of processor A are equal while least significant bits of previous and current address of processor B are not), the FCU proceeds to step 570. At step 570, the FCU 20 locks processor A and allows processor B to access the memory at step 570. If processor B caused the jump, the FCU locks processor B while allowing processor A to access the memory at step 560.

A situation may occur where both processors performed 25 a jump. In such a case, the FCU proceeds to step 580 and

examines a priority register which contains the information indicating which processor has priority. In one embodiment, the priority register is toggled to alternate the priority between the processors. As shown in Fig. 5, 5 the FCU toggles the priority register at step 580 prior to determining which processor has priority. Alternatively, the priority register can be toggled after priority has been determined. In one embodiment, a 1 in the priority register indicates that processor A has priority (step 585) 10 while a 0 indicates that processor B has priority (step 590). Using a 1 to indicate that B has priority and a 0 to indicate that A has priority is also useful. The same process can also be performed in the event a conflict occurred in which neither processor performed a jump (e.g., 15 least significant bits of the current and previous addresses of processor A or of processor B are not the same).

In alternative embodiments, other types of arbitration schemes can be also be employed by the FCU to synchronize 20 the processors. In one embodiment, the processors may be assigned a specific priority level vis-à-vis the other processor or processors.

Fig. 6 shows a block diagram of a portion of a system 600 in accordance with one embodiment of the invention. 25 The system comprises, for example, multiple digital signal

processors (DSPs) for multi-port digital subscriber line (DSL) applications on a single chip. The system comprises  $m$  processors 610, where  $m$  is a whole number equal to or greater than 2. Illustratively, the system comprises first 5 and second processors 610a-b ( $m = 2$ ). Providing more than two processors is also useful.

A memory module 660 is provided for sharing among the processors. Data words accessed by the processors are stored in the memory module. A data word comprises a group 10 of bits (e.g. 32 bits). In one embodiment, the data words comprise program instructions, which are accessed by the processors from the memory module via memory buses (e.g. 618a and 618b) for execution. The data words can also comprise application data.

15 In accordance with one embodiment of the invention, the memory module is shared between the processors without noticeable performance degradation, eliminating the need to provide duplicate memory modules for each processor. Noticeable performance degradation is avoided by separating 20 the memory module into  $n$  number of independently operable banks (e.g. 665a and 665b), where  $n$  is a number greater than or equal to 2. Preferably,  $n =$  the number of processors in the system (i.e.  $n = m$ ). Since the memory banks operate independently, the different banks can be 25 simultaneously accessed during the same clock cycle.

In another embodiment, the banks can be further subdivided into x number of independently accessible blocks 675a-p, where x is an integer greater than or equal to 1. A bank, for example, is subdivided into 8 independently 5 accessible blocks. Generally, the greater the number of blocks, the lower the probability of contention. The number of blocks, in one embodiment, is selected to optimize performance and reduce contention.

The blocks of the memory array have, for example, 10 control circuitry 668 to appropriately place data on the memory buses (e.g. 618a or 618b) to the processors (610a or 610b). The control circuitry comprises, for example, multiplexing circuitry or tri-state buffers to direct the data to the respective processors. By providing 15 independent blocks within a bank, the processors can advantageously access different blocks simultaneously, irrespective of whether they are from the same bank or not. This further increases system performance by reducing potential conflicts between processors.

Furthermore, the memory is mapped so that contiguous 20 memory addresses are rotated between the different memory banks. For example, in a two-bank memory module (e.g., bank 0 and bank 1), one bank (bank 0) would be assigned the even addresses while odd addresses are assigned to the 25 other bank (bank 1). This would result in data words in

sequential addresses being located in alternate memory banks, such as data word 1 in bank 0, data word 2 in bank 1, data word 3 in bank 0 and so forth. In one embodiment, the data words comprise program instructions. Since 5 program instructions are executed in sequence with the exception of jumps (e.g., branch and loop instructions), a processor would generally access different banks of the memory module during program execution. By synchronizing or staggering the processors to execute the program so that 10 the processors access different memory banks in the same cycle, multiple processors can execute the same program stored in memory module 660 simultaneously.

An arbitration control unit (ACU) 645 being coupled to the processor via the data bus and to the memory module via 15 the memory bus is provided. The ACU controls access to the memory by the processors. In the event of a memory contention (e.g., two processors accessing the same bank simultaneously), the ACU determines which processor has priority to access the memory module while the other 20 processors are locked (e.g. by executing a wait state or cycle). This generally synchronizes the processors to access different banks in the subsequent clock cycles.

In one embodiment, a priority register is provided to indicate which processor has priority. In the case of a 25 system with two processors, the priority register may

comprise one bit (P bit). Additional bits may be included to accommodate additional number of processors. The priority register is updated after the occurrence of contention to rotate the priority between the processors.

5 For example, a value of '1' in the P bit indicates that the first processor has priority and a '0' indicates that the second processor has priority. During each cycle where a contention occurs, the P bit is toggled, switching the priority of the processors. Other types of arbitration  
10 schemes are also useful.

Optionally, the processors can be provided with respective critical memory modules 615. The critical memory module, for example, is smaller than the main memory module 660 and is used for storing programs or subroutines  
15 which are accessed frequently by the processors (e.g., MIPS critical). The use of critical memory modules enhances system performance by reducing memory conflicts without going to the extent of significantly increasing chip size.

The ACU 645 is coupled to n control logic units  
20 (CLUs), one for each of the n processors. Illustratively, the ACU comprises first CLU 648a and second CLU 648b for first processor 610a and second processor 610b respectively. When a CLU is activated, its respective processor is allowed access to the memory module. In one  
25 embodiment, the CLU is coupled to a processor and to the n

banks of memory module, enabling the processor to access the n memory banks simultaneously. Since the bandwidth of a processor is equal to the bandwidth of a memory bank, the CLU allows the processor to fetch from memory more words than needed. In one embodiment, the processor can potentially fetch twice the data words needed.

In one embodiment, the CLU comprises first (cache) and second (normal) signal paths. The cache signal path comprises, for example, a cache register (633a or 633b) and a multiplexer (636a or 636b). When the cache path is selected, the processor coupled to the CLU accesses the first and second memory banks (665a-b). In one embodiment, the current address location (Addr), as specified by the processor, and the next address (Addr + 1) are accessed. The multiplexer selects the word at (Addr + 1) and stores it in the cache while the word at the current address (Addr) is passed to the processor. The address of the word stored in the cache is stored in, for example, a cache address register (640a or 640b). If the second path (normal) is selected, the processor accesses the current memory location. The CLU passes the data word at the current memory location to the processor via the second path. By providing a cache to store data in subsequent addresses, the probability of data access from memory is

lowered, hence reducing memory latency caused by memory contention.

The processors can be provided with respective critical memory modules 615a and 615b. The critical memory module, for example, is smaller than the main memory module 660 and is used for storing data (e.g. programs or subroutines) which are accessed frequently by the processors (e.g., MIPS critical). The use of critical memory modules enhances system performance by reducing memory conflicts without going to the extent of significantly increasing chip size.

Fig. 7 shows a process flow of an ACU state machine in accordance with one embodiment of the invention. As shown, the ACU controls accesses by first and second processors (A or B). The ACU system is initialized (710), for example, before system operation (e.g., system power up). Initialization includes, for example, setting the priority bit to indicate which processor has priority in the event of a memory contention. The priority register, for example, is set to give processor A priority.

During operation of the system, the processors issue respective memory addresses corresponding to the memory access in the next clock cycle ( $A_{Addr}$  and  $B_{Addr}$  representing the memory addresses currently issued by processor A and processor B). The ACU determines whether there is a memory

contention or not at steps 720 and 722, e.g., whether the processors are accessing the same memory range or not. The memory range coincides, in one embodiment, with a memory block. In another embodiment, the memory range coincides 5 with memory blocks in different banks, the memory blocks comprising consecutive addresses. If no contention exists, processors A and B access respective banks of the memory module at step 750. In one embodiment, the CLUs of processors A and B are activated with the normal signal 10 paths selected. Thus, each processor retrieves data words from respective memory banks at addresses  $A_{Addr}$  and  $B_{Addr}$ .

If a contention occurs, the ACU evaluates the priority register to determine which processor has access priority at step 726. The processor P with access priority (e.g., 15 processor A) is allowed access to the memory while the other processor P' with lower priority executes a wait state (e.g., processor B) at step 728. Hence, if the processors subsequently access data words in sequential locations in the next cycles, different banks will be 20 accessed without executing wait-states. By synchronizing or staggering the processors to execute the program so that the processors access different memory banks in the same cycle, multiple processors can execute the same program stored in memory module 660 simultaneously without 25 contention.

In one embodiment, the CLU of processor P is activated with the cache signal path selected, at step 730. The data from the current address  $P_{Addr}$  and the next consecutive address  $P_{Addr+1}$  are fetched from the memory banks. The data 5 in the current address  $P_{Addr}$  is passed to the processor P for access and data in the next address  $P_{Addr+1}$  is stored in the cache register. The ACU updates the priority at step 332 for the next contention evaluation at step 722.

The ACU determines at step 734 if a new address  $P_{Addr}$  10 specified by the processor P in the next cycle matches the address of the cache data (i.e. cache hit). If a cache miss occurs, the process is repeated by evaluating the addresses specified by processors A and B for contention at step 720. In one embodiment, the data in the cache 15 register associated with processor P is discarded.

A cache hit would allow processor P to continue execution by retrieving the data from the cache instead of memory, thus avoiding the insertion of a wait-state at step 736. In one embodiment, the CLU of processor P' is 20 activated with the cache signal path selected at step 734. The data from the current address  $P'_{Addr}$  and the next address  $P'_{Addr+1}$  are fetched from the memory banks. The data in the current address  $P'_{Addr}$  is passed to the processor P' for access and the data in the next address  $P'_{Addr+1}$  is 25 stored in the cache register associated with P'. If there

is a cache hit for processor P' in the next cycle, the cache data is accessed by the processor P' at step 740. The data in the current address  $P_{Addr}$  of processor P accessed by the processor and the data in the next address 5  $P_{Addr+1}$  is stored in the cache register associated with P. There is no need to check for contention as only one processor is accessing the memory. The determination of a cache hit for processor P is repeated at step 734. If a cache miss for P' occurs at step 738, the ACU repeats the 10 whole process at step.

In another embodiment shown in Fig. 8, the ACU comprises the cache signal path for each processor, the cache signal path allowing more data words to be fetched from memory than requested by the processor. The cache 15 signal path comprises, for example, a cache register and a multiplexer. During the operation of the system, the addresses issued by processors A and B are evaluated for contention at steps 820 and 822. If contention exists, the priority is evaluated and wait states are inserted for the 20 processor with lower priority as previously described in Fig. 7.

If no contention exists, the caches associated with both processors are evaluated for cache hit at step 852. If no cache hits are found, processors A and B access 25 respective banks of the memory module at step 850 via the

respective cache signal paths. In one embodiment, the CLUs of processors A and B are activated with the cache paths selected. The data in the current memory addresses ( $A_{Addr}$  and  $B_{Addr}$ ) is passed to the respective processors for access 5 and data in the next consecutive addresses ( $A_{Addr+1}$  and  $B_{Addr+1}$ ) is stored in the respective cache registers. If cache hits are detected for both processors, the respective cache contents are accessed by the processors at step 862 and the process repeats at step 820.

- 10 If a cache hit is found for only one of the processors, memory access may continue for the other processor without the need to test for contention since only one processor is accessing the memory. For example, if a cache hit is detected for processor A and a cache miss 15 is detected for processor B, the contents of the cache associated with processor A is accessed while the data from the current memory address  $B_{Addr}$  is accessed by processor B at step 854. Data from the memory at the next location  $B_{Addr+1}$  is stored in the cache associated with processor B.
- 20 In the next cycle, the cache for processor B is monitored again for a cache hit. If a cache hit occurs, the cache contents for processor B is retrieved at step 856. The data from memory at address  $A_{Addr}$  will be fetched for processor A. A cache miss at step 858 will cause the 25 process to be repeated from step 820.

Figs. 9-10 illustrate the mapping of memory in accordance with different embodiments of the invention.

Referring to Fig. 9, a memory module 260 with 2 banks (Bank 0 and Bank 1) each subdivided into 8 blocks (Blocks 0-7) is shown. Illustratively, assuming that the memory module comprises 512Kb of memory with a width of 16 bits, each block being allocated 2K addressable locations ( $2K \times 16$  bits  $\times$  16 blocks). In one embodiment, even addresses are allocated to bank 0 (i.e., 0, 2, 4 ... 32K-2) and odd addresses to bank 1 (i.e., 1, 3, 5...32K-1). Block 0 of bank 0 would have addresses 0, 2, 4 ... 4K-2; block 1 of bank 1 would have addresses 1, 3, 5...4K-1.

Referring to Fig. 10, a memory module with 4 banks (Banks 0-3) each subdivided into 8 blocks (Blocks 0-7) is shown. Assuming that the memory module 512Kb of memory with a width of 16 bits, than each block is allocated 1K addressable locations ( $1K \times 16$  bits  $\times$  32 blocks). In the case where the memory module comprises 4 banks, as shown in Fig. 10, the addresses would be allocated as follows:

20 Bank 0: every fourth address from 0 (i.e., 0, 4, 8,  
etc.)

Bank 1: every fourth address from 1 (i.e., 1, 5, 9,  
etc.)

25 Bank 2: every fourth address from 2 (i.e., 2, 6, 10,  
etc.)

Bank 3: every fourth address from 3 (i.e., 3, 7, 11,  
etc.)

The memory mapping can be generalized for n banks as  
follows:

5 Bank 0: every  $n^{\text{th}}$  address beginning with 0 (i.e., 0,  
n, 2n, 3n, etc.)

Bank 1: every  $n^{\text{th}}$  address beginning with 1 (i.e., 1,  
1+n, 1+2n, 1+3n, etc.)

.

10 .

.

Bank  $n-1$ : every  $n^{\text{th}}$  address beginning with  $n-1$  (i.e.,  
 $n-1, n-1+n, n-1+2n, \text{ etc.}$ )

While the invention has been particularly shown and  
15 described with reference to various embodiments, it will be  
recognized by those skilled in the art that modifications  
and changes may be made to the present invention without  
departing from the spirit and scope thereof. The scope of  
the invention should therefore be determined not with  
20 reference to the above description but with reference to  
the appended claims along with their full scope of  
equivalents.

What is claimed is:

1. A method of sharing a memory module between a plurality of processors comprising:

dividing the memory module into n banks, where n = at least 2, wherein each bank can be accessed by one or more processors at any one time;

mapping the memory module to allocate sequential addresses to alternate banks of the memory; and

storing data bytes in memory, wherein said data bytes in sequential addresses are stored in alternate banks due to the mapping of the memory.

2. The method of claim 1 further including a step of

dividing each bank into x blocks, where x = at least 1,

wherein each block can be accessed by one of the plurality of processors at any one time.

3. The method of claim 1 or 2 further including a step of determining whether memory access conflict has occurred,

wherein two or more processors are accessing the same block at any one time.

4. The method of claim 1, 2 or 3 further including a step of synchronizing the processors to access different blocks at any one time.

5. The method of claim 4 further including a step of determining access priorities of the processors when memory access conflict occurs.

5 6. The method of claim 5 wherein the step of determining access priorities comprises assigning lower access priorities to processors that have caused the memory conflict.

10 7. The method of claim 5 wherein the step of determining access priorities comprises assigning lower access priorities to processors that performed a jump.

15 8. The method of claim 4, 5, 6 or 7 wherein the step of synchronizing the processors comprises locking processors with lower priorities for one or more cycles when memory access conflict occurs.

20 9. A system comprising:  
a plurality of processors;  
a memory module comprising n banks, where n = at least 2, wherein each bank can be accessed by one or more processors at any one time;  
a memory map for allocating sequential addresses to alternate banks of the memory module; and

data bytes stored in memory, wherein said data bytes in sequential addresses are stored in alternate banks according to the memory map.

5 10. The system of claim 9 wherein each bank comprises  $x$  blocks, where  $x =$  at least 1, wherein each block can be accessed by one of the plurality of processors at any one time.

10 11. The system of claim 9 or 10 further comprising a flow control unit for synchronizing the processors to access different blocks at any one time.

15 12. The system of claim 9, 10 or 11 further comprising a priority register for storing the access priority of each processor.

13. The system of any of claims 9-12 wherein said data bytes comprise program instructions.

20

14. The system of any of claims 10-13 further comprising a plurality of critical memory modules for storing a plurality of data bytes for each processor for reducing memory access conflicts.

25

15. A method of sharing a memory module between a plurality of processors comprising:

dividing the memory module into n banks, where n = at least 2, enabling the memory module to be accessed by one  
5 or more processors simultaneously;

mapping the memory module to allocate sequential addresses to alternate banks of the memory;

storing data words in memory, wherein data words in sequential addresses are stored in alternate banks due to  
10 the mapping of the memory; and

providing a first signal path, the first signal path coupling a cache to a processor and the memory module when selected, the cache enabling the processor to fetch a plurality of data words from different banks  
15 simultaneously.

16. The method of claim 15 further including a step of dividing the bank into x blocks, where x = at least 1, wherein a block can be accessed by one of the plurality of  
20 processors at any one time.

17. The method of claims 15 or 16 further including a step of determining whether contention has occurred, wherein two or more processors are accessing the same address range at  
25 any one time.

18. The method of claim 17 wherein the address range coincides with at least one block.

19. The method of any of the claims 15-18 further  
5 including a step of synchronizing the processors to access different banks when contention has occurred.

20. The method of any of the claims 15-19 further including the step of providing a second signal path, the  
10 second signal path coupling the processor to the memory module when selected.

21. The method of any of the claims 15-20 further including a step of activating the second signal path when  
15 contention has not occurred.

22. The method of any of the claims 15-21 further including a step of synchronizing the processors to access different banks when contention has occurred.

20

23. The method of any of the claims 15-22 further including a step of determining access priorities of the processors when contention has occurred.

24. The method of claim 23 wherein the step of determining access priorities comprises assigning lower access priorities to processors that have caused the contention.

5 25. The method of any of the claims 19-24 wherein the step of synchronizing the processors comprises inserting wait states for processors with lower priorities when contention occurs.

10 26. The method of any of the claims 15-25 further including a step of activating the first signal path when contention has occurred.

27. A system comprising:

15       a plurality of processors;  
            a memory module comprising n banks, where n = at least 2, wherein a bank can be accessed by one or more processors at any one time;

20       a memory map for allocating sequential addresses to alternate banks of the memory module;

            data words stored in memory, wherein data words in sequential addresses are stored in alternate banks according to the memory map; and

a plurality of control logic unit for enabling a processor to access a plurality of data words from different banks.

5 28. The system of claim 27 wherein a control logic unit comprises first and second signal paths, the first signal path coupling a cache to a processor and the memory module, the second signal path coupling the processor to the memory module.

10

29. The system of claim 27 or 28 wherein the first signal path comprises a cache register and a multiplexer.

30. The system of any of the claims 27-29 wherein the bank  
15 comprises x blocks, where x = at least 1, wherein a block can be accessed by one of the plurality of processors at any one time.

31. The system of any of the claims 27-30 further comprising a flow control unit for synchronizing the  
20 processors to access different blocks at any one time.

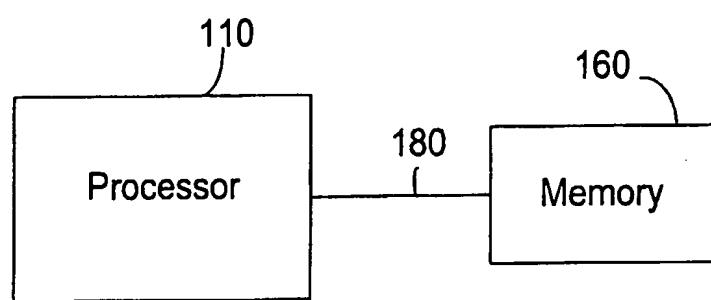
32. The system of any of the claims 27-31 further comprising a priority register for storing the access  
25 priority of a processor.

33. The system of any of the claims 27-32 further comprising a plurality of critical memory modules for storing a plurality of data words for the processors to reduce the possibility of contention.

5

34. The system of any of the claims 27-33 wherein a control logic unit comprises a first signal path, the first signal path coupling a cache to a processor and the memory module.

1/10



100

FIG 1

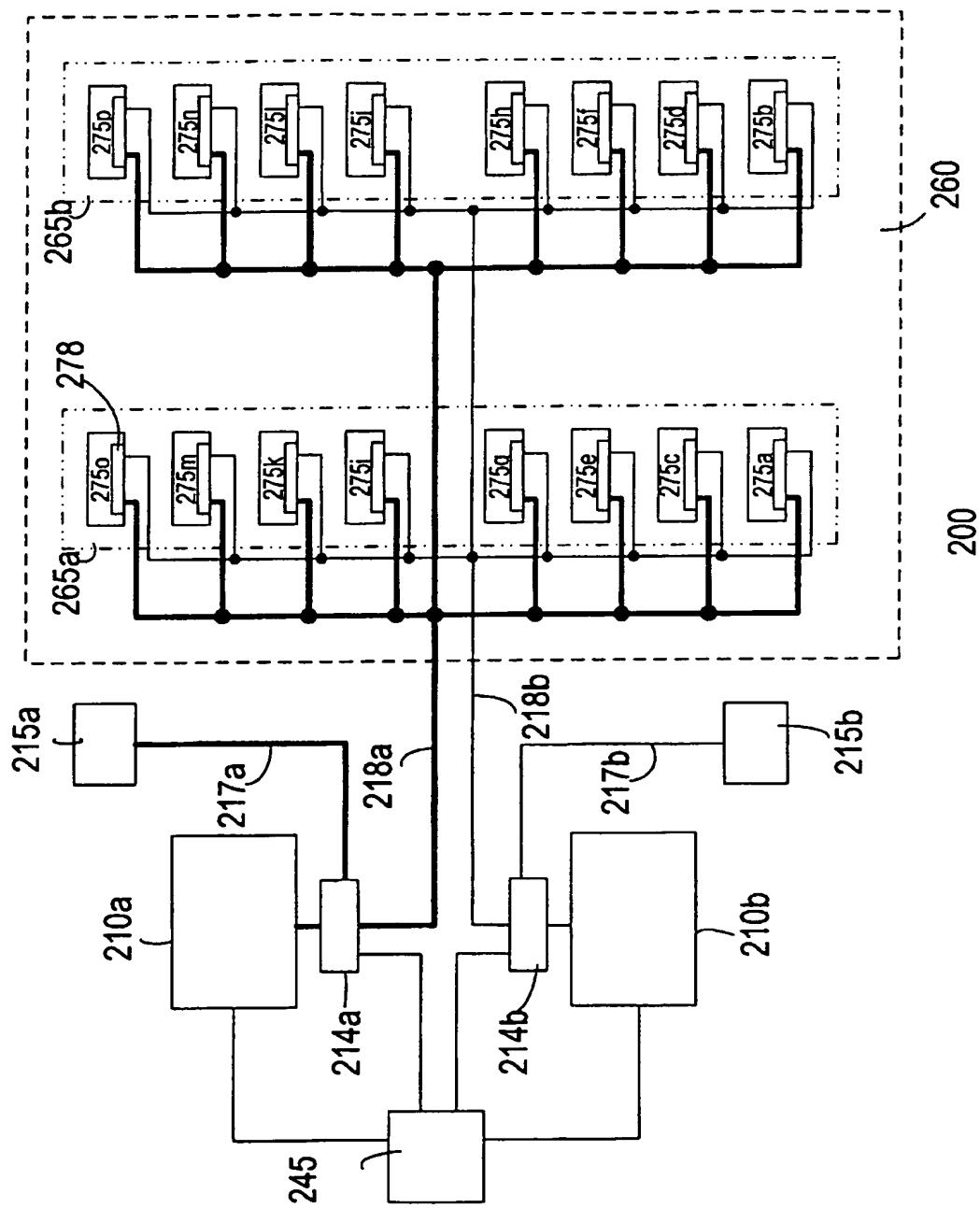
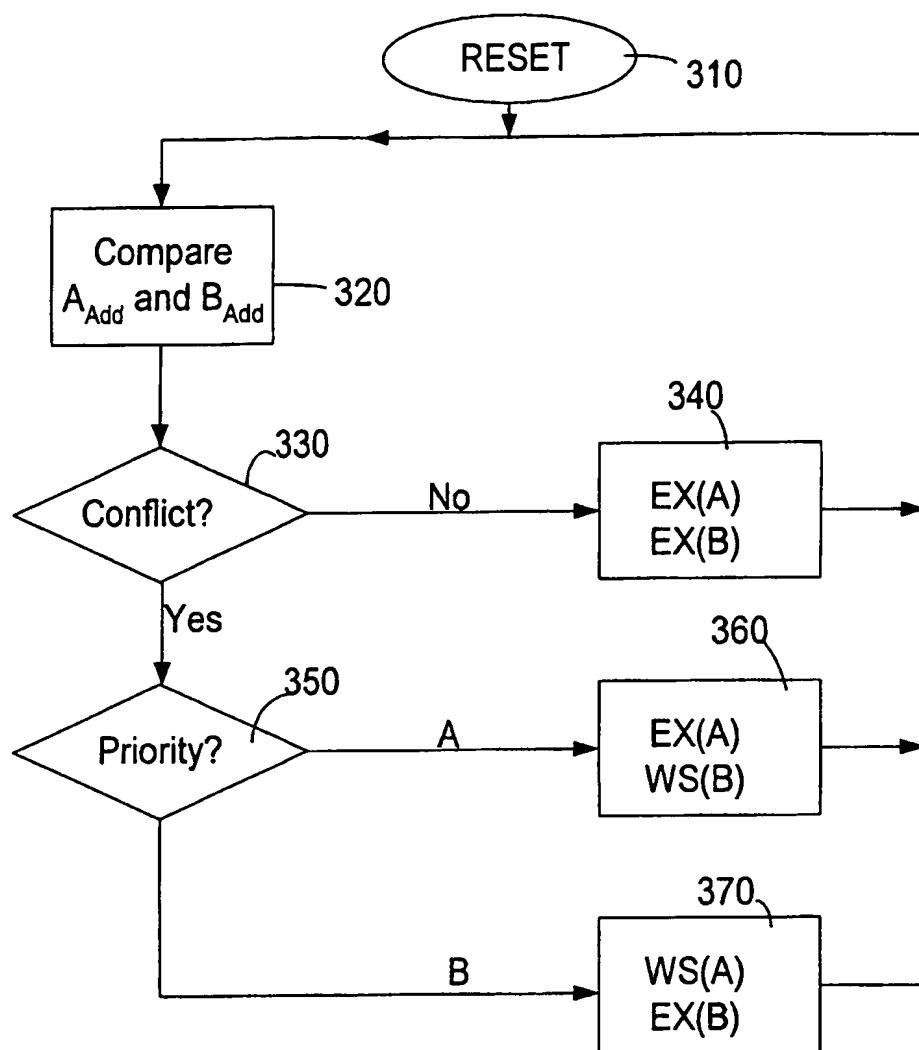


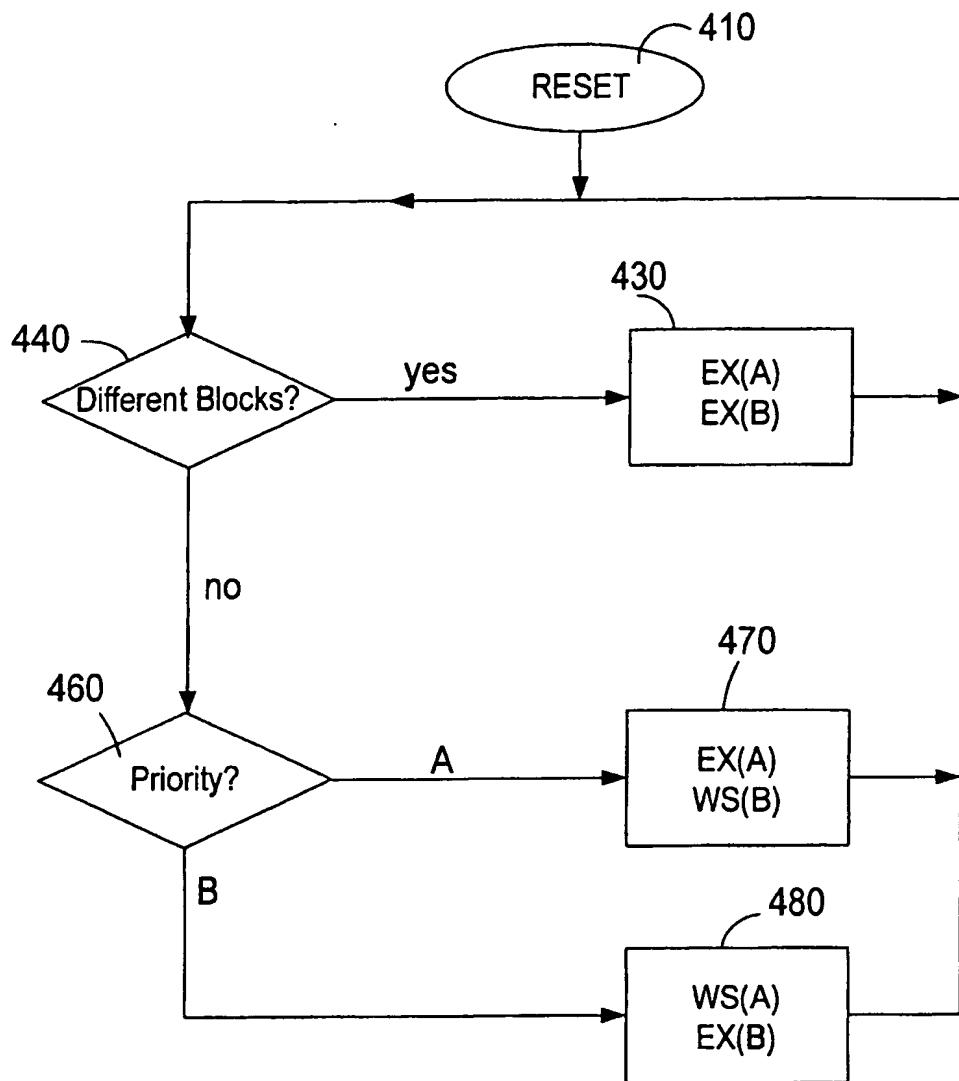
FIG 2

3/10



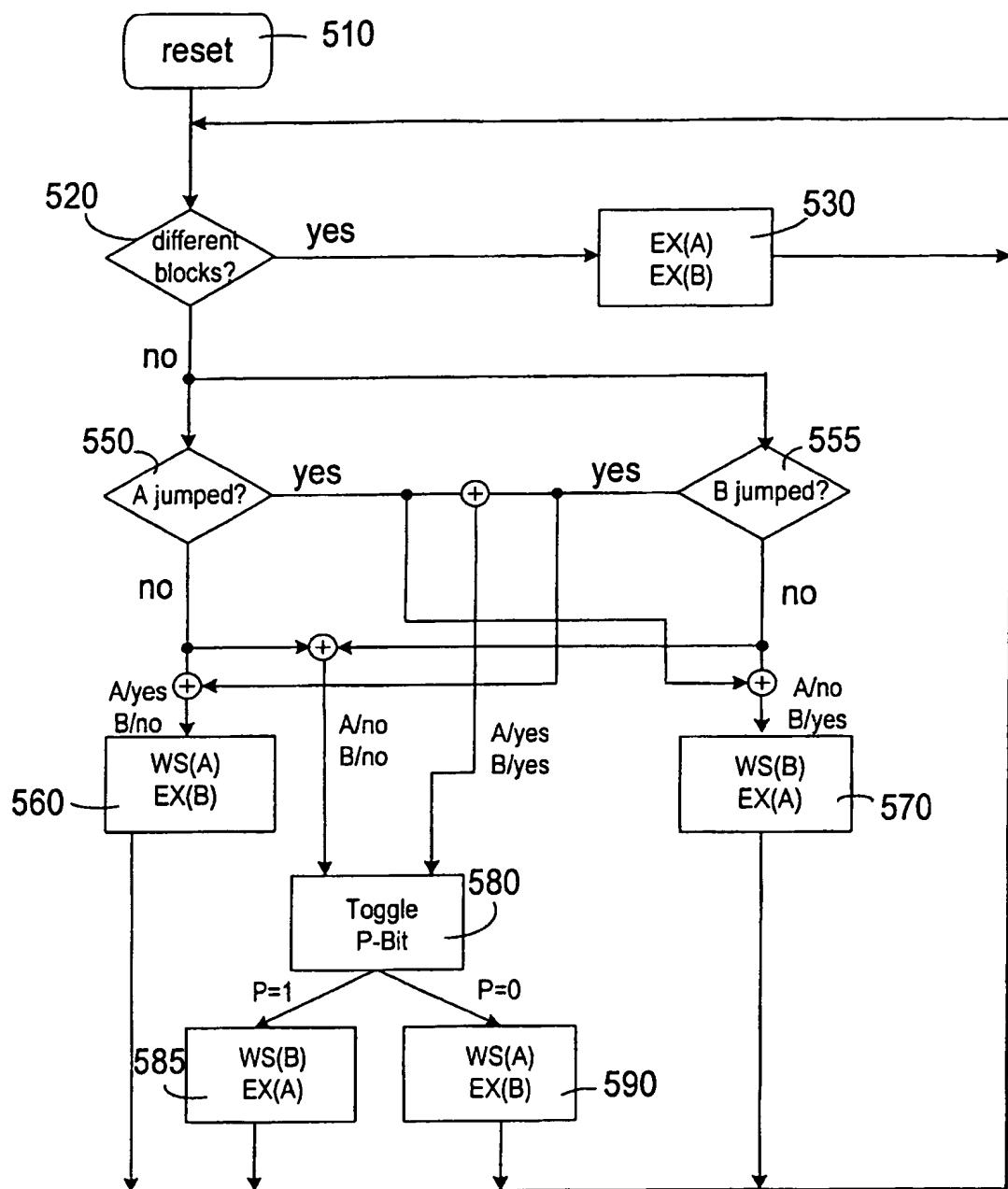
301

FIG 3



401

FIG 4



501

FIG 5

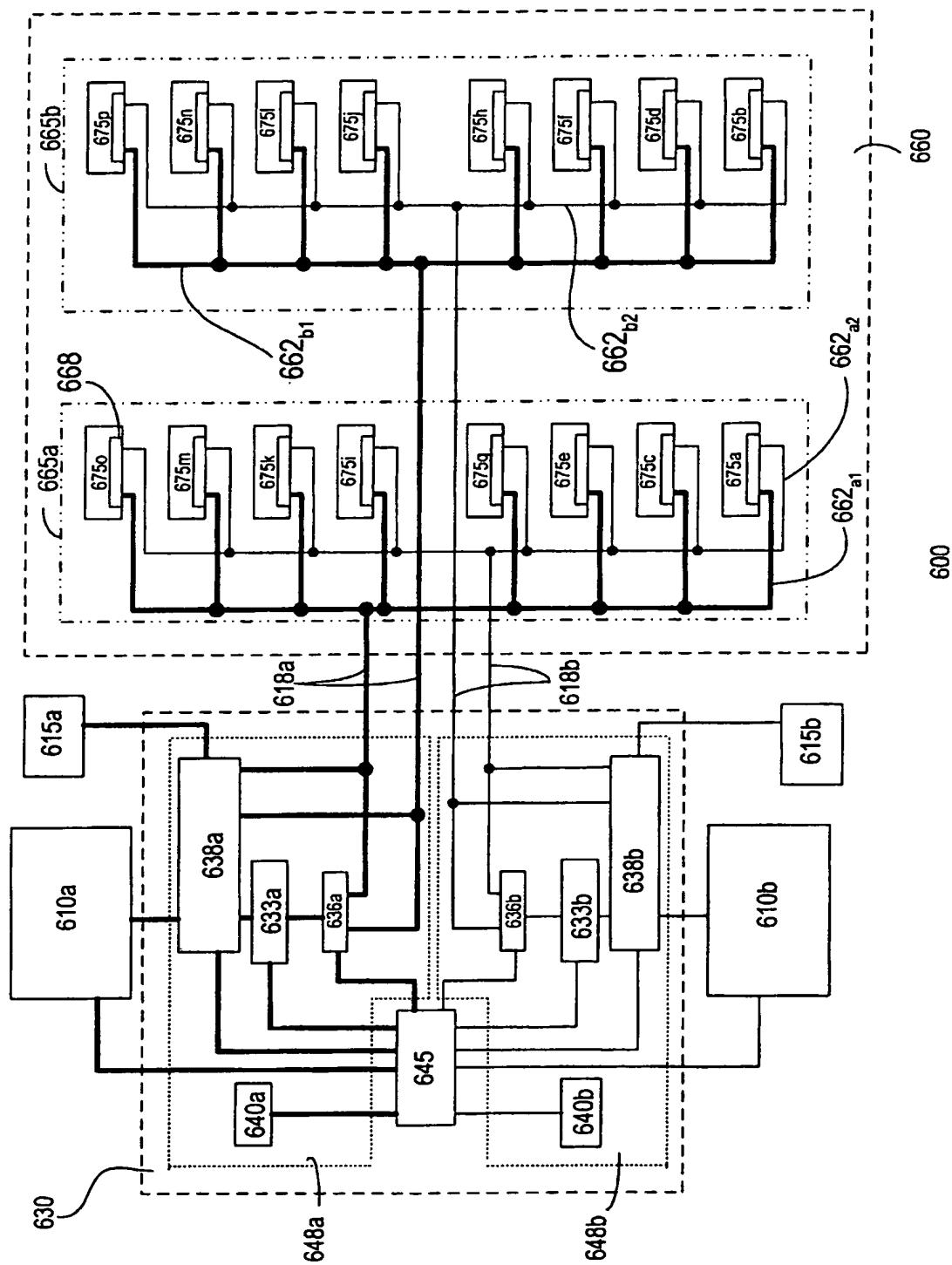


FIG 6

7 / 10

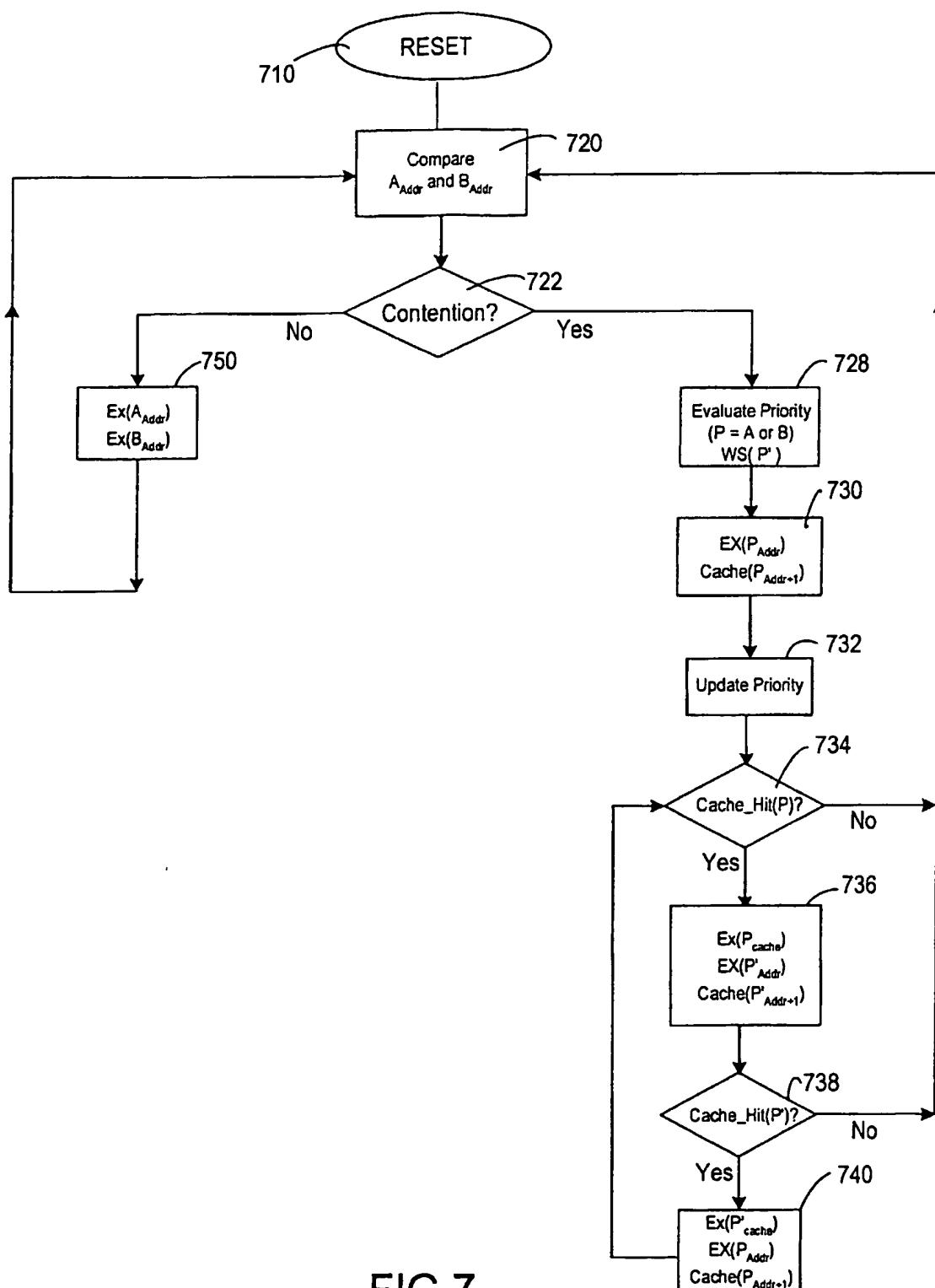


FIG 7

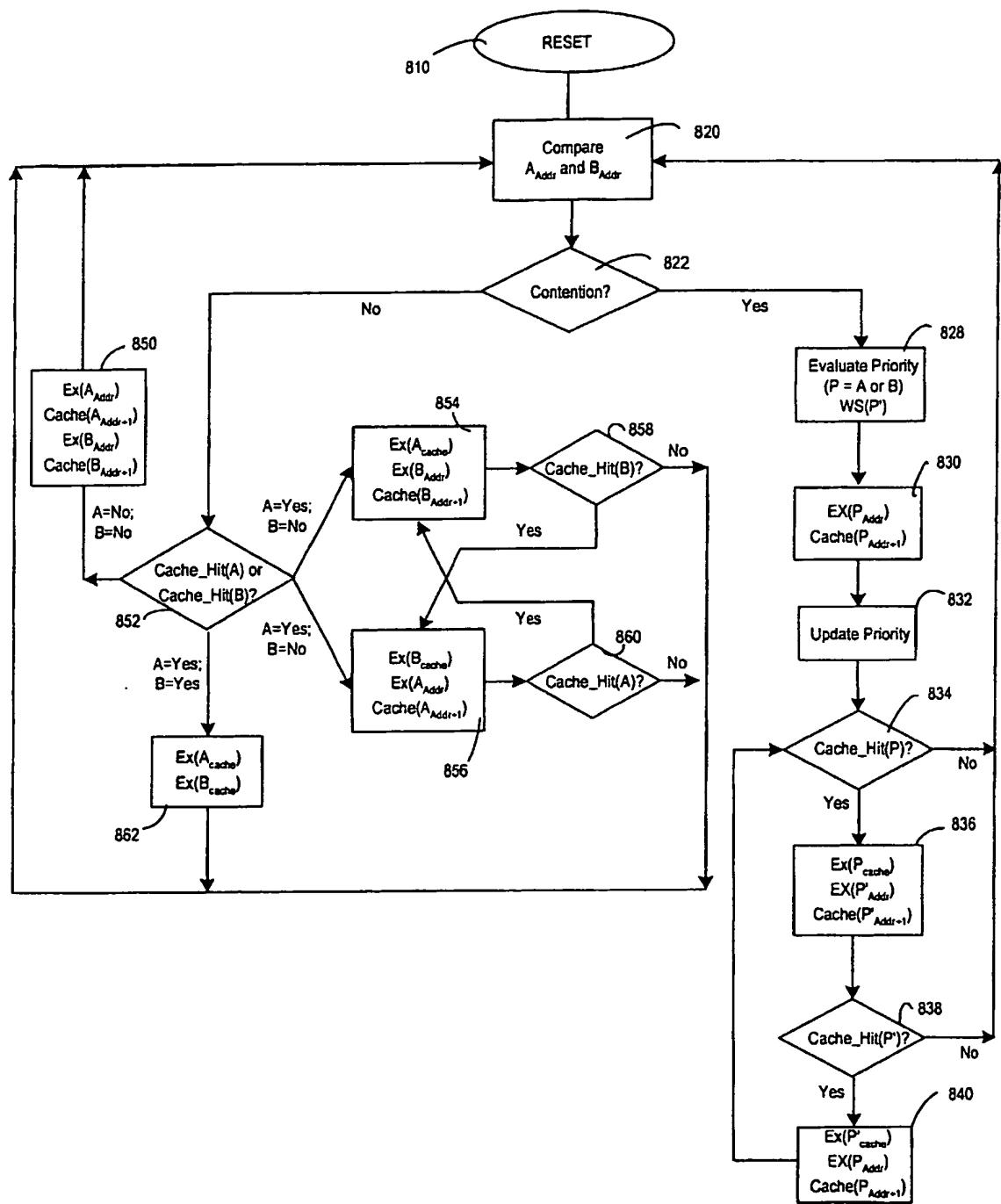


FIG 8

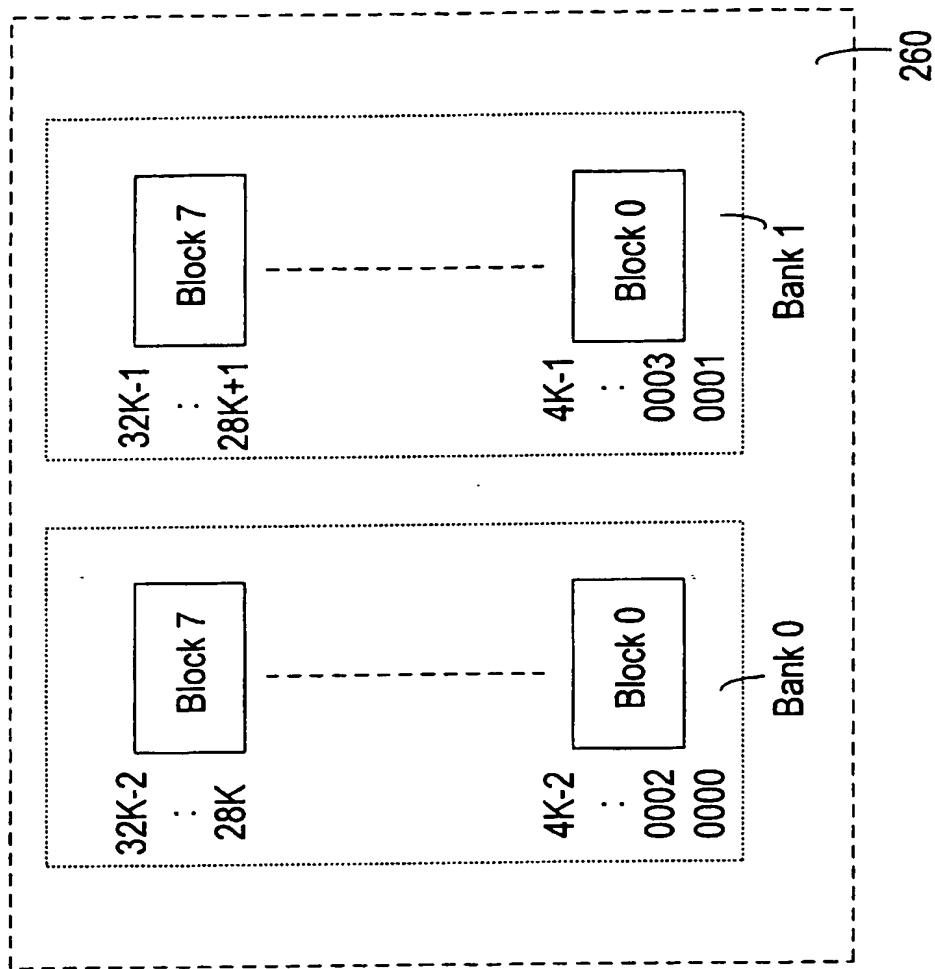


FIG 9

10/10

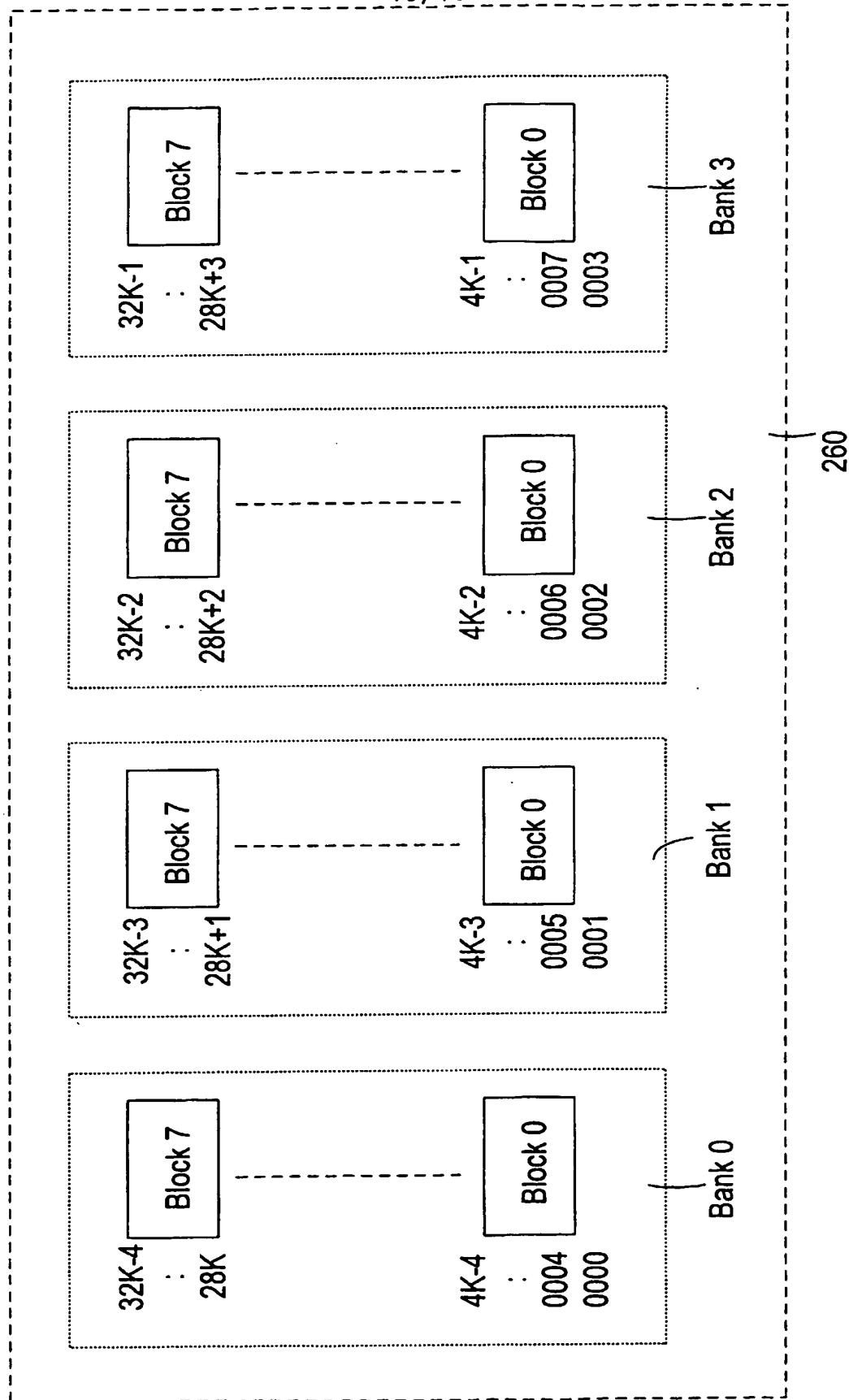


FIG 10



## INTERNATIONAL SEARCH REPORT

International Application No

PCT/EP 03/03547

## A. CLASSIFICATION OF SUBJECT MATTER

IPC 7 G06F9/46 G06F12/06

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, INSPEC, COMPENDEX, PAJ

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 3 931 613 A (GRUNER RONALD H ET AL) 6 January 1976 (1976-01-06)  column 5, line 18 - line 23 column 5, line 36 -column 6, line 2 column 10, line 6 - line 33 column 11, line 54 - line 61 column 12, line 51 - line 60 figure 11	1-5, 9-11,13, 14
Y		12, 15-20, 22,23, 25,27-34
Y	US 5 857 110 A (HASHIMOTO SHIGEKO ET AL) 5 January 1999 (1999-01-05) column 11, line 62 -column 12, line 7 figures 2,3 ----	12,32 ---- -/--

 Further documents are listed in the continuation of box C. Patent family members are listed in annex.

° Special categories of cited documents :

- "A" document defining the general state of the art which is not considered to be of particular relevance
- "E" earlier document but published on or after the international filing date
- "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- "O" document referring to an oral disclosure, use, exhibition or other means
- "P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the International filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search

12 January 2004

Date of mailing of the International search report

01 06. 2004

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel: (+31-70) 340-2040, Tx. 31 651 epo nl.  
Fax: (+31-70) 340-3016

Authorized officer

de Man, A

## INTERNATIONAL SEARCH REPORT

International Application No

PCT/EP 03/03547

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	<p>US 4 918 587 A (PECHTER RICHARD G ET AL) 17 April 1990 (1990-04-17)</p> <p>column 2, line 64 -column 3, line 14 column 3, line 27 - line 38 column 4, line 47 - line 51 column 6, line 9-12 figure 1</p> <p>-----</p>	15-20, 22,23, 25,27-34
E	<p>WO 03 041119 A (TERSCHLUSE MARKUS ;FRENZEL RUDI (DE); HORAK CHRISTIAN (DE); UHLEMA) 15 May 2003 (2003-05-15)</p> <p>the whole document</p> <p>-----</p>	1-14

## INTERNATIONAL SEARCH REPORT

International application No.  
PCT/EP 03/03547

### Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1.  Claims Nos.: because they relate to subject matter not required to be searched by this Authority, namely:
  
2.  Claims Nos.: because they relate to parts of the International Application that do not comply with the prescribed requirements to such an extent that no meaningful International Search can be carried out, specifically:
  
3.  Claims Nos.: because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

### Box II Observations where unity of Invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see additional sheet

1.  As all required additional search fees were timely paid by the applicant, this International Search Report covers all searchable claims.
  
2.  As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
  
3.  As only some of the required additional search fees were timely paid by the applicant, this International Search Report covers only those claims for which fees were paid, specifically claims Nos.:
  
4.  No required additional search fees were timely paid by the applicant. Consequently, this International Search Report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

#### Remark on Protest

- The additional search fees were accompanied by the applicant's protest.
- No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. Claims: 1-14

Method of memory interleaving in a multi-processor system wherein an access conflict is resolved by assigning a lower priority to processors that caused the conflict.

2. Claims: 15-34

Method of memory interleaving in a multi-processing system wherein a cache is used to enable the processor to fetch a plurality of data words from different memory banks simultaneously.

**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International Application No

PCT/EP 03/03547

Patent document cited in search report		Publication date		Patent family member(s)		Publication date
US 3931613	A	06-01-1976		AU 8511875 A CA 1040743 A1 CA 1049153 A2 DE 2542010 A1 DE 2560630 A1 FR 2286439 A1 GB 1520485 A JP 984670 C JP 51051253 A JP 54016384 B NL 7511289 A		31-03-1977 17-10-1978 20-02-1979 15-04-1976 04-07-1985 23-04-1976 09-08-1978 22-01-1980 06-05-1976 21-06-1979 29-03-1976
US 5857110	A	05-01-1999		JP 3260456 B2 JP 6195313 A US 5617575 A JP 3413843 B2 JP 5165716 A US 5392443 A		25-02-2002 15-07-1994 01-04-1997 09-06-2003 02-07-1993 21-02-1995
US 4918587	A	17-04-1990		DE 3887324 D1 DE 3887324 T2 EP 0345325 A1 JP 2502495 T JP 2835757 B2 WO 8905488 A2		03-03-1994 08-09-1994 13-12-1989 09-08-1990 14-12-1998 15-06-1989
WO 03041119	A	15-05-2003		US 2003088744 A1 WO 03040900 A2 WO 03041119 A2 WO 03041010 A2 WO 03040915 A1 US 2003088801 A1 US 2003085742 A1 US 2003086487 A1 WO 03085524 A2		08-05-2003 15-05-2003 15-05-2003 15-05-2003 15-05-2003 08-05-2003 08-05-2003 08-05-2003 16-10-2003

DT09 Rec'd CT/PTO 13 SEP 2008

10/507408

The PTO did not receive the following  
listed item(s) No Post Card